

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平11-272454

(43)公開日 平成11年(1999)10月8日

(51) Int.Cl.⁶

G O 6 F 9/06

識別記号

4 1 0

FI

G O 6 F 9/06

4 1 0 Q

審査請求 未請求 請求項の数13 O L (全 20 頁)

(21)出願番号 特願平11-6203

(22)出願日 平成11年(1999)1月13日

(31)優先権主張番号 9801661.1

(32)優先日 1998年1月28日

(33)優先権主張国 イギリス (GB)

(71)出願人 390009531

インターナショナル・ビジネス・マシーンズ・コーポレーション

INTERNATIONAL BUSINESS
MACHINES CORPORATION

アメリカ合衆国10504、ニューヨーク州
アーモンク (番地なし)

(72)発明者 シーマス・ドノヒュー

アイルランド共和国、ダブリン・5、アル
 タン、セント・ジョーンズ・コート 34

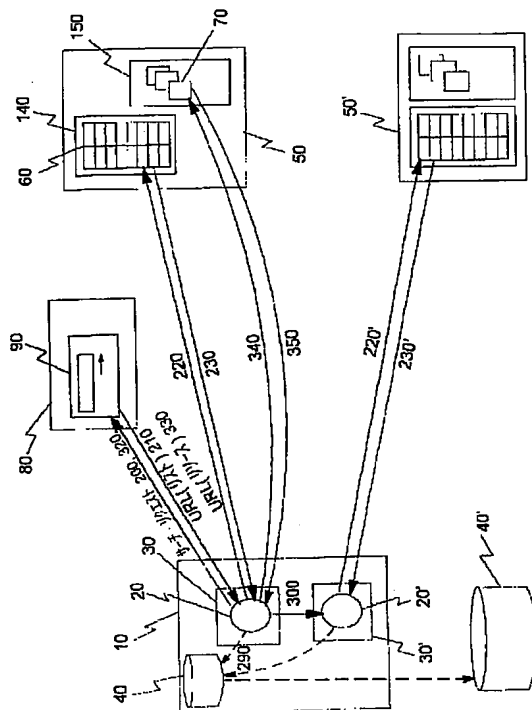
(74)代理人 弁理士 坂口 博 (外1名)

(54) 【発明の名称】 コンピュータ・ネットワークを通してソフトウェア更新を配布する装置

(57) 【要約】

【課題】 コンピュータ・プログラムの更新を自動化するための方法及び機構を提供する。

【解決手段】通常、コンピュータ・プログラムは、ユーザが自分のコンピュータ・システム上に導入するために記録媒体を通して配布される。プログラムに関する補修、追加、及び新バージョンが行われる度に、ユーザがその更新を導入することを可能にするために、新しいCD又はディスクがユーザに配布される必要がある。最近では、或るソフトウェアはネットワークを介してダウンロード可能になったが、ユーザが更新情報を得てそれらを導入する労力、及びソフトウェア・ベンダが更新情報を分配するための労力は望ましくないままである。本発明は、コンピュータ・プログラムと関連付けられたアップデータ・エージェントを提供する。



【特許請求の範囲】

【請求項1】 コンピュータ・ネットワーク内で接続されたコンピュータ・システム上に導入された1つ又は複数のコンピュータ・プログラムを更新する場合に使用するためのアップデータ・コンポーネントにして、

1つ又は複数の必要なソフトウェア・リソースを検索するために、前記1つ又は複数の必要なソフトウェア・リソースが設けられた前記ネットワーク内の1つ又は複数の識別可能なロケーションへのアクセスを開始するための手段と、

1つ又は複数の検索されたソフトウェア・リソースを使用して前記導入されたコンピュータ・プログラムの1つにソフトウェア更新を施すための手段と、
を含むアップデータ・コンポーネント。

【請求項2】 使用可能な関連の更新リソースを識別するために、前記1つ又は複数の識別可能なロケーションから得られるソフトウェア更新リソースと前記コンピュータ・システム上に導入された1つ又は複数のコンピュータ・プログラムとの比較を行い、前記使用可能な関連の更新リソースと前記コンピュータ・システムにおいて記憶された事前定義の更新基準とを比較するための手段と、

ソフトウェア・リソースを自動的にダウンロードし、前記事前定義の更新基準を満たすソフトウェア更新を施すための手段と、

を含む請求項1に記載のアップデータ・コンポーネント。

【請求項3】 前記事前定義の更新基準は適用可能なソフトウェア製品ユーザ・ライセンスによる許容し得る更新の範囲の定義を含む請求項2に記載のアップデータ・コンポーネント。

【請求項4】 前記ソフトウェア更新を施すための手段は前記事前定義の更新基準に従って及び更新のためにダウンロードされたソフトウェア・リソースの一部である導入のためのコンピュータ読み取り可能な命令に従って、使用可能な関連のソフトウェア・リソースを導入するための手段を含む請求項2又は請求項3に記載のアップデータ・コンポーネント。

【請求項5】 1つ又は複数のロケーションを識別するための情報が前記アップデータ・コンポーネントによって保持され、コンピュータ・プログラム製品の製品識別子を含み、
前記アップデータ・識別子は前記製品識別子をサーチ・エンジンに供給するように適応し、
前記製品識別子は前記サーチ・エンジンがネットワーク・ロケーションを識別するために使用するためのサーチ・パラメータとして働く、

請求項1乃至請求項4の何れかに記載のアップデータ・コンポーネント。

【請求項6】 前記アップデータ・コンポーネントは、

使用可能なソフトウェア更新リソースのリストが保持されているネットワーク・ロケーションを前記サーチ・エンジンが識別することに応答して前記リスト及び前記リソースの前提ソフトウェア製品をダウンロードし、

前記リスト及び前提ソフトウェア製品と前記コンピュータ・システム上に導入されたコンピュータ・プログラムとを比較し、

前記前提ソフトウェア製品に対する更新が必要である場合に前記前提ソフトウェア製品に対する更新をリクエストするように適応する請求項5に記載のアップデータ・コンポーネント。

【請求項7】 前記アップデータ・コンポーネントはコンピュータ・システム上に前記アップデータ・コンポーネントを導入するための機械読み取り可能な導入命令を有し、

前記導入命令は前記アップデータ・コンポーネントが他のアップデータ・コンポーネントによって識別可能及び接触可能であるように他のアップデータ・コンポーネントによってアクセスし得るリポジトリによって前記アップデータ・コンポーネントを登録するための命令を含む、

請求項1乃至請求項6の何れかに記載のアップデータ・コンポーネント。

【請求項8】 前記アップデータ・コンポーネントは、現在のアップデータ・コンポーネントがそのコンピュータ・プログラムを更新することを相補的なコンピュータ・プログラムがリクエストする時に介するAPIを含み、

前記現在のアップデータ・コンポーネントは更新リクエストに応答してそのコンピュータ・プログラムを更新するために更新方法呼び出すように適応し、

前記現在のアップデータ・コンポーネントは、そのコンピュータ・プログラムが前提コンピュータ・プログラムの更新を必要とする時、システム発生されたリクエストをそのコンピュータ・プログラムの前記前提コンピュータ・プログラムのアップデータ・コンポーネントに送るように適応する請求項6又は請求項7に記載のアップデータ・コンポーネント。

【請求項9】 前記更新を施すための手段は存在する導入済みのソフトウェアを修正する訂正及び機能拡張ソフトウェアを導入し、

導入されたソフトウェアを置換する導入済みのソフトウェアのアップグレードしたバージョンを導入するように適応する請求項1乃至請求項8の何れかに記載のアップデータ・コンポーネント。

【請求項10】 コンピュータ読み取り可能な記録媒体上に記録されたコンピュータ・プログラム・コードを含み、

前記コンピュータ・プログラム・コードは前記コンピュータ・プログラム・コードを更新するための請求項1乃至請求項9の何れかに記載のアップデータ・コンポーネント。

至請求項 9 の何れかに記載された統合アップデータ・コンポーネントを含むコンピュータ・プログラム製品。

【請求項 1 1】コンピュータ読み取り可能な記録媒体上のレコードのためのコンピュータ・プログラム・コードを含み、

前記コンピュータ・プログラム・コードは前記コンピュータ・プログラム・コードを更新するための請求項 1 乃至請求項 9 の何れかに従って統合アップデータ・コンポーネントを含むコンピュータ・プログラム製品。

【請求項 1 2】コンピュータ・ネットワーク内で接続されたコンピュータ・システム上に導入されたコンピュータ・プログラムを自動的に更新するための方法にして、前記コンピュータ・プログラムを更新する場合に使用するためのアップデータ・コンポーネントを前記コンピュータ・システムに配布するステップと、前記コンピュータ・プログラムを現在のバージョンから更新済みのバージョンに形成するためのダウンロード可能なソフトウェア・リソースを第 1 ネットワーク・ロケーションに提供するステップと、前記コンピュータ・システムにおいて実行される時、前記アップデータ・コンポーネントが遂行するように適応するステップであって、

(a) 前記アップデータ・コンポーネントによって保持された情報から識別可能であり、又は前記アップデータ・コンポーネントによってアクセス可能であって、前記ソフトウェア・リソースが配置される前記第 1 ネットワーク・ロケーションへのアクセスを開始するステップと、

(b) 前記ソフトウェア・リソースを前記コンピュータ・システム上にダウンロードするステップと、

(c) ダウンロードされたソフトウェア・リソースを使用して前記コンピュータ・プログラムを前記現在のバージョンから前記更新済みのバージョンに更新するステップと、を含む方法。

【請求項 1 3】前記アップデータ・コンポーネントにおける情報から識別可能な第 2 ネットワーク・ロケーションに前記コンピュータ・プログラムにとって使用可能な更新のコンピュータ読み取り可能なリストを設けるステップを含み、

前記アップデータ・コンポーネントによって、前記第 1 ネットワーク・ロケーションにアクセスする前に遂行されるように適応するステップにして、

前記リストを検索するために前記第 2 ネットワーク・ロケーションへのアクセスを開始するステップと、

使用可能な関連の更新リソースを識別するために、前記リストを読み取り、リストされた使用可能な更新と前記第 1 コンピュータ・システム上の前記コンピュータ・プログラムとの比較を行うステップと、

前記使用可能な関連の更新リソースと前記アップデータ

・コンポーネントにおける事前定義された更新基準とを比較し、前記更新基準を満たす更新のための使用可能な関連の更新リソースを識別するステップと、を含む請求項 1 2 に記載の方法。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】本発明は、コンピュータ・ネットワークを介してソフトウェアを配布すること及びコンピュータ・ネットワークを介してソフトウェア機能強化、訂正、又は新バージョンの情報をアクセスするための機構に関するものである。コンピュータの「ネットワーク」は、相互に情報を交換することができる任意の数のコンピュータであってもよく、或いは任意の構成で配列され、任意の接続方法を使用するものでよい。

【0 0 0 2】

【従来の技術】通常、ソフトウェアは、ディスクット又はコンパクト・ディスクのような記録媒体上に記録されたプログラムの形式で配布されている。顧客は、記録媒体及びその媒体上に記録されたソフトウェアを使用するライセンスを買い、しかる後、その記録媒体から自分のコンピュータ上にそのソフトウェアを導入する。事前記録された記録媒体の製造及び配布は高価なものであり、このコストは顧客に課せられるであろう。又、顧客がソフトウェアを注文又は購入すべく努力することは望ましいことではない。

【0 0 0 3】ほとんどのソフトウェアは、そのソフトウェアがユーザに送達された後、バグを訂正するために及び新たな特徴的機能を追加するために頻繁に更新されるので、その配布コストが特に問題である。或るタイプのソフトウェア製品は毎年何回も更新される。ソフトウェアがアップグレードされ又は訂正される度にすべての登録メンバに新しいディスクット又は CD を送るコストは禁止的に高価であり、しかも、多くの顧客は自分のソフトウェアが最新ののものであって最高パフォーマンスのバージョンであること、及びエラーのないものであることを望んでいるけれども、すべての顧客があらゆる更新を受けることを望んでいるわけではない。例えば、ベンダは顧客が使うことを望んでいるものよりも多くのものを更新に対して課することがあり、或いは、新しいバージョンは顧客が買うことを望んでいない他の前提ソフトウェア製品のアップグレードを要求することがあり、或いは、新しいバージョンに移行することは或る期間の間顧客のシステムを不能にするデータのマイグレーションを必要とすることがある。

【0 0 0 4】従って、ソフトウェア・ベンダは、自分のソフトウェアの新しいバージョンの可用性を公表し、最新のアップグレードされたバージョンを購入すべきかどうかの決定を顧客に委ねる傾向がある。しかし、或るソフトウェア製品に関しては、ソフトウェア・ベンダが、自分のソフトウェア製品に対するアップグレードされた

バージョン又は少なくともエラー訂正及び機能強化コード（「パッチ」として知られている）を先行学習を目的に送り出すことが適している。特定の企業のポリシーがどのようなものであろうと、これらの種々なタイプのソフトウェア更新のリリースにはかなりのコストと労力が伴う。

【0005】益々、ソフトウェア流通業者は、自分のソフトウェアに対する更新の可用性を公表するための機構として、更に或るソフトウェアを配布ための機構としてインターネットを使用しようとしている。インターネットは、単一のオーナー又はコントローラを持たない大型及び小型の公衆及び私設ネットワークを含むコンピュータ・ネットワークのネットワークであり、そこでは、インターネット・プロトコル・ソフトウェアを走らせる如何なる接続されたコンピュータもセキュリティ・コントロールを受け、インターネットに接続された他の任意のコンピュータと情報を交換することができる。相互に接続することに同意したネットワークのこの複合集合体は単一の伝送媒体に依存するのではない（例えば、双方向通信は衛星回線、光ファイバ中継回線、電話回線、ケーブル・テレビ線、及びローカル無線回線を介して生じ得る）。

【0006】ワールド・ワイド・ウェブ・インターネット・サービス（以下では、「ウェブ」と呼ぶ）は、極めて多数のネットワーク・アクセス可能な情報へのアクセスを提供し、インターネット接続されたコンピュータ相互間の低コスト通信を提供し得る広域情報検索機能である。インターネット・アクセスを有するソフトウェア・ベンダの顧客が製品の入手可能な最新バージョンをリストを手操作でチェックし、しかる後、それらの製品をオンラインで注文することが知られている。これは、ソフトウェアを注文する場合に伴う書類事務作業の量を減少させる（しかも、これは他の製品に同様に適用可能である）。或る企業は自分のソフトウェアがサーバ・コンピュータにおけるウェブ・サイトから顧客自身のコンピュータに直接にダウンロードされることも可能にしている（しかし、このダウンロード機能は、セキュリティ上の理由で及びパッチを適用することが前提ソフトウェアに対する如何なる変更も又は如何なるデータ・マイグレーションも必要としない傾向があるため、バグ修復パッチ、低コスト・プログラム、及びプログラムのデモ・コピー又は評価コピーに限定されることが多い）。

【0007】ワールド・ワイド・ウェブに関する情報は、Andrew Ford 氏による「ウェブをスピニング（Spinning the Web）」と題した文献（ロンドンの International Thomson Publishing 社により 1995 年発行）、及び John December 及び Neil Randall 氏による「解放されたワールド・ワイド・ウェブ（The World Wide Web Unleashed）」と題した文献（インディアナポリスの SAMS Publishing 社より 1994 年発行）において見る

ことができる。WWWの使用は、それと融通性、携帯性、及び使用の容易性との結合のため、対話式マルチメディア・プレゼンテーション機能と結合されて爆発的な速度で成長しつつある。WWWは、インターネットに接続され且つ適正なソフトウェア及びハードウェア構成を有する任意のコンピュータが、インターネット上のどこかで利用可能にされた任意のドキュメントを検索することを可能にする。

【0008】ソフトウェアの注文及び配布のためのインターネットのこのような使用の増加はソフトウェア・ベンダのためのコストを節約したが、多くのソフトウェア製品に関して、ベンダは、自分のウェブ・ページを適切な時間にすべての顧客がアクセスするという完全に依存することはできず、従って、付加的な更新機構が望ましい。

【0009】製造コスト及び配布媒体と関連した配布コストの問題の他に、顧客がソフトウェア製品の最良の及び最新のバージョン及びリリースを有するかどうかを知るために、及び更新を得てそれらの更新を施すために、一般には、顧客はかなりの先行学習を目的に努力を行う必要があるという問題が存在する。この努力は、インターネットの接続が得られる時には少なくなるけれども、ウェブ・サイトの先行学習を目的にしたチェックを必要とすることさえも多くのユーザにとって望ましいことではない。これは、それがチェックを行うための注意点を設定すること、ソフトウェア提供者のウェブ・サイトを見つけてアクセスすること、最新のソフトウェア・バージョン及びパッチがリストされているウェブ・ページにナビゲートすること、及び関連製品の更新が得られるかどうかを決定するために及びそれが注文されるべきかどうかを決定するために、導入済みのソフトウェアとこのリスト内のバージョン及びリリース番号とを比較することを伴うためである。更新を注文することとそれが使用のために利用可能になることとの間に望ましくない遅延が存在することがあり、しかも、たとえその更新が直ちにダウンロードされ得るものであっても、ソフトウェア製品のアップグレードされたバージョンに移行するというタスクは困難であることがある。これらのステップは、アプリケーション、コントロール・パネル、拡張子、ユーティリティ、及びシステム上にインストールされたシステム・ソフトウェア・プログラム毎に繰り返されなければならない場合、更新は非常に冗長となり、時間浪費的なものになる。従って、手操作の更新は十分に又は規則的に遂行されない傾向がある。

【0010】ソフトウェア・ベンダは、自分のソフトウェアのどのバージョンが各顧客によって使用されているかを知らないと言う関連の問題がある。たとえば、それらのソフトウェアの最新のバージョンがすべての登録された顧客に精力的に（CDを送出することによって又はサーバ制御のオンライン配布によって）配布さ

れたとしても、顧客が更新の正しい導入の面倒を見ると
いう保証は依然としてない。これはソフトウェア開発者
の或る自由を取り去る。なぜならば、ソフトウェア開発
者は、一般に、自分のソフトウェアの前のバージョンと
逆互換性を維持しなければならないか、或いはアップグ
レードしないユーザに対して別の譲歩を行わなければな
らないためである。

【0011】クライアント／サーバ・コンピューティン
グ環境では、サーバ側におけるシステム・アドミニスト
レータが自分の判断でソフトウェア製品の新しいバージ
ョンをクライアント・システムにおけるエンド・ユーザ
に課することが知られている。しかし、これはアドミニ
ストレータがクライアントのシステムを更新するための
アクセス制御を有する場合に可能であったにすぎない。
これは、アップグレードが課せられることを望まないユ
ーザを考慮するものではない。

【0012】更なる関連の問題として、ソフトウェア製
品は、それが働くことを可能にするために他のソフトウ
ェア製品を必要とすることが多いということがある。例
えば、アプリケーション・プログラムは、一般に、特定
のオペレーティング・システムに対して書かれる。1つの
製品の特定のバージョンは他の製品の特定のバージ
ョンを必要とすることが多いので、他の製品をアップグ
レードすることなく第1の製品をアップグレードすること
は、その結果として第1の製品が働かなくなることがあ
る。

【0013】「インサイダ更新2.0 (Insider Updates
2.0)」は、インサイダ・ソフトウェア社 (Insider So
ftware Corporation) から商業的に入手可能なソフトウ
ェア・アップデート・ユーティリティである。それは、
ユーザによってトリガされる時、ユーザのアップル・マ
ッキントッシュ・コンピュータにおける導入済みソフト
ウェアのインベントリを作成し、これと利用可能なソフト
ウェア更新パッチ (しかし、アップグレードされた製
品バージョンではない) のデータベースとを比較し、関
連の更新をダウンロードする。「インサイダ更新」は関
連の更新を見つけるための責任をユーザからデータベ
ースの保守側に移すが、更新パッチへのアクセスは個々の
データベースへの接続に限定され、更新を見つけるため
にインターネット及びオンライン・サービスをスキャン
するタスク及び利用可能な更新のデータベースを維持す
るタスクはかなりの先行学習を目的に努力することを必
要とする。「インサイダ更新」は更新を導入せず、或い
はユーザのソフトウェアを如何なる方法でも修正しな
い。「インサイダ更新」は非同期化された前提ソフトウ
ェア製品の問題を処理しない。

【0014】導入されたソフトウェアを決定するために
コンピュータ・システムの選択されたボリュームをスキ
ャンし、アップル・マッキントッシュに対するソフトウ
ェア・タイトルのデータベースに接続するが更新をダウ

ンロードしない同様の製品は、シンメトリ・ソフトウ
ェア社 (Symmetry Software Corporation) の「バージョ
ン・マスタ1.5 (Version Master 1.5)」である。

【0015】別の更新方法がシャーマン社 (Shaman Cor
poration) からの「シャーマン更新サーバ1.1.6 (Sh
aman Update Server 1.1.6)」によって提供される。そ
れは、ユーザがPower Macファイル・サーバ上に
導入するCD-ROM (毎月更新され、配布される)
と、インベントリ登録され、更新されるべき各マッキ
ントッシュ・コンピュータのためのクライアント・ソフト
ウェアと、現在の更新のライブラリを記憶するFTPサ
イトをアクセスするための手段とより成る。「シャーマ
ン更新サーバ」は、ネットワーク結合されたコンピュ
ータのインベントリを作成し、ソフトウェアの最新バージ
ョンを各コンピュータにダウンロード及び配布する。ネ
ットワーク・アドミニストレータはこのインベントリ及
び更新プロセスを中心的に制御する。CD-ROMの配
布は前述の費用の問題を有する。

【0016】

【発明が解決しようとする課題】本発明の第1の局面に
よれば、コンピュータ・ネットワーク内で接続されたコ
ンピュータ・システム上に導入された1つ又は複数のコ
ンピュータ・プログラムを更新する場合に使用するため
のアップデート・コンポーネントが提供される。そのア
ップデータ・コンポーネントは、1つ又は複数の必要な
ソフトウェア・リソースが置かれているそのネットワ
ーク内の1つ又は複数のロケーションを識別するための情
報、前記1つ又は複数の必要なソフトウェア・リソース
を検索するために前記1つ又は複数のロケーションへの
アクセスを開始するための手段、及び前記1つ又は複数
の検索されたソフトウェア・リソースを使用して前記導
入されたコンピュータ・プログラムの1つにソフトウ
ェア更新を施すための手段を含む。

【0017】本発明によるアップデート・コンポーネ
ントは、関連するソフトウェア製品のアップグレード及び
その製品におけるバグの修復を、更新基準に関する初期
の同意後、ユーザによる如何なる対話も必要とすること
なく自動的に制御することが望ましい。更新基準は、製
品のライセンシング条件と関連付けることが可能であ
る。これは、エラーがユーザの観点から自動的に訂正さ
れる場合、適切な更新ポリシーを選んだユーザがいつも最
も最新の利用可能なソフトウェアを持つことができるこ
とを保証する。ユーザは、アップデート・コンポーネ
ントがこれを処理するので、ソフトウェア更新が生じる場
所、それらを得る方法、又はそれらを導入する方法を知
る必要がない。ソフトウェア・ベンダは、エラーを訂正
するために特別のCD又はディスクットを出荷する必要
がなくなるし、更なる特徴を与える必要もなくなる。ベ
ンダは、顧客が新しい製品の特徴を早く及び努力するこ
となく受けるように、漸次的増加を基準にして容易にコ

ードをリリースすることができる。

【0018】本発明の望ましい実施例によるアップデート・コンポーネントはローカル・コンピュータ・システムにおける利用可能なソフトウェア更新と導入済みソフトウェアとの間の比較を行い、どれが導入されたソフトウェアと関連しているかを識別し、利用可能な関連の更新とローカル・コンピュータ・システム上に保持された更新基準（これらの更新基準は現在のシステム又はシステム・ユーザに対して事前定義される）とを比較し、しかる後、その事前定義された基準を満足するソフトウェア更新を自動的にダウンロードし、その更新を施す。

【0019】このように、ソフトウェア更新を自動的に施すことは、事前定義された更新基準と、更新を必要とするプログラム・コードと共にダウンロードされる導入のための命令との両方に従って、利用可能なソフトウェア・パッチ及び／又はアップグレードされたバージョンを導入することを伴う。ダウンロードされた命令を動的に実行するという特徴は、アップデート・コンポーネントにより処理可能な更新のタイプに関連して融通性を与える。それは、単一の汎用アップデート・コンポーネントを多くの種々のソフトウェア製品と共に使用することを可能にするためにも使用可能である。別の方法として、或るソフトウェア更新のための導入命令を、アップデート・コンポーネント内で事前符号化してもよい。その「ソフトウェア・リソース」は、一般に、プログラム・コード、機械読み取り可能な導入命令、及びアドレス情報のような任意の必要なデータ変更の組合せである。

【0020】ネットワーク・ロケーションを識別する場合に使用するための情報は明示的ネットワーク・ロケーション情報であってよく、或いは、それはソフトウェア・ベンダ名又はそのロケーションを識別するためのサーチ・パラメータとして使用可能な他の任意の情報であってもよい。望ましい実施例では、その情報は、その製品に更新を施すためのソフトウェア・リソースを記憶された関連のネットワーク・ロケーションを識別するためのサーチを開始するために、アップデート・コンポーネントによってサーチ・エンジンに供給される製品識別子である。このサーチは、アップデート・コンポーネントによって呼び出される通常のインターネット（又は他のネットワーク）サーチ・エンジンによって遂行可能である。サーチ・エンジンがネットワーク・ロケーションの識別情報を戻す時、アップデート・コンポーネントは利用可能な関連の更新のリストをこのロケーションから検索し、ローカル的に保持されたソフトウェア製品バージョン及び事前定義された更新基準に関してそのリストをチェックし、それらの基準が満たされる場合、更新リソースを検索してローカル・コンピュータ・システム上に与える。

【0021】本発明の望ましい実施例によれば、ソフトウェア更新を形成すべきソフトウェア・リソースに対し

て、標準化されたネーミングの常套手段が使用され、しかもアップデート・コンポーネントは、ネットワーク・オペレーティング・システムのファイル・システムにおいてこれらのリソースに関してサーチすることができる。これは、ネットワーク可用性問題を緩和するためにソフトウェア・リソースが複数のロケーションに記憶されることを可能にし、開発者及び配布者がそれらのエラー修復パッチ及びソフトウェア製品のアップグレードされたバージョンを提供することを容易にする。例えば、開発者は、それらのLANにおける既知のファイル名を使用した公衆ネットワーク・ディスク・ドライブを介して、又は既知のキー・ワードを使用するためにサーチ可能な公開されたユニフォーム・リソース・ロケータ（URL）を介して得られる新しいソフトウェア更新を作ることができる。

【0022】アップデート・コンポーネントは、それらが更新するために働く製品の不可欠な部分であることが望ましい。従って、アップデート・コンポーネントは、ソフトウェア製品の初期バージョンと共にソフトウェア・ユーザに配布される。しかる後、アップデート・コンポーネントは、プリセット基準（更新に関する相次ぐサーチの相互間の期間のような、及び特定のユーザがすべての更新を受けることを選択したか、或いは、例えば、パッチの更新を受けるが置換製品バージョンを受けないというような或る更新だけを受けることを選択したか）に従ってソフトウェア更新を自動的に得て、それを施す。

【0023】アップデート・コンポーネントの更新機能は更新そのものを含むことが望ましい。実に、アップデート・コンポーネントが、その関連のソフトウェア製品を更新するためのソフトウェア・リソースを求めてそれがサーチする前に、それ自身に対する更新を得るために適正なネットワーク・ロケーションをいつもアクセスするように、更新基準はセット可能である。

【0024】本発明によるアップデート・コンポーネントは、前提製品が利用可能であるかどうかをチェックするための手段を含むことが望ましく、しかも、現在の製品に対する更新パスを選択するプロセスの部分として、必要なバージョンに同期化されることが望ましい。望ましい実施例では、それらの可用性をチェックすること以外に、アップデート・コンポーネントは、これが同意済みの更新ポリシーである場合、前提ソフトウェア製品と関連したアップデート・コンポーネントにそれらのソフトウェアに対する更新を開始するように命令することができる。各ソフトウェア製品のアップデート・コンポーネントが前提製品に対する更新をトリガすることができる場合、更新は、ユーザがその更新に関与することなく又はその更新を知ることを必要とすることなく、導入されたソフトウェア製品のセットを通して波及することができる。この機能は、更新が行われる時に非同期ソフトウ

エア・バージョンの問題を扱わない従来のアップデータ・エージェントに比べて大きな利点であり、エンド・ユーザのためのタスクを遂行するために配布オブジェクト相互間のコラボレーションに関してソフトウェア業界における増加する動向を支援する。

【0025】アップデータ・コンポーネントは、暗号アルゴリズムを使用して、ダウンロードされたソフトウェアの真偽を検査するための機構を含むことも望ましいことである。これは、専用の、パスワード保護された、及び別の方法で保護されたソフトウェア・リソース・リポジトリ・サイトを必要ないものにする。ソフトウェア・リソースは、それらが正しく名前を付けられ、ネットワーク・サーチ・エンジンに通知される限り、ネットワーク上のどこにあってもよい。

【0026】従って、本発明は、ソフトウェア更新を得るための及びその更新を施すためのエージェント及び方法を提供する。この方法は、ソフトウェア更新を配布及び追跡するソフトウェア・ディストリビュータに関するコスト及び労力を大幅に減らし、導入されたソフトウェアに更新を施すシステム・アドミニストレータ及びエンド・ユーザに関する労力を大幅に減らす。

【0027】

【課題を解決するための手段】図1に示されるように、アップデータ・コンポーネント20が、通常のネットワーク接続されたコンピュータ・システム10のシステム・メモリに、関連のコンピュータ・プログラム30と共に導入される。そのアップデータ・コンポーネントは、ローカル・コンピュータ・システムのユーザが導入するために、記憶媒体（ディスク又はCD）によってユーザに配布されてもよく、或いは他のコンピュータ・システムから明示的にダウンロードされてもよい。本発明の望ましい実施例では、アップデータ・コンポーネントは、それらが保守することを意図された（又は、それとは別に、同じ機構を介して及びそれらが関連したプログラムと同時に配布される）コンピュータ・プログラム内に統合される。アップデータ・コンポーネントは、ユーザがそれを取得し又は活性化するために如何なる特別のアクションを取ることも必要ないように、その関連のプログラムの導入手順の一部として導入される。各アップデータ・コンポーネントの導入は、そのアップデータ・コンポーネントがオペレーティング・システムと共にそれ自身を登録することを含み（更に一般的には、アップデータ・コンポーネントは、中心的な又は分散したリポジトリ40を登録し）、従って、少なくともローカル・システムにおけるアップデータ・コンポーネントは、レジスタ・エントリにおけるアドレス情報及び／又はそれらの製品識別子によって識別可能であり、制御可能である。

【0028】本発明の望ましい実施例の特徴は、各アップデータ・コンポーネントが、他のソフトウェア製品を

管理する他のアップデータ・コンポーネントを見つけることができ、他のアップデータ・コンポーネントによって見つけられ、しかも他のアップデータ・コンポーネントとコミュニケーションすることができるということである。この機能は、一方のアップデータ・コンポーネントが他方のアップデータ・コンポーネントを、前者が後述のようにそれ自身の更新を実行する前に特定のレベルに更新するために必要とする時使用される。これは、アップデータ・コンポーネントがオペレーティング・システム又は他のリポジトリ40内に登録することによって可能にされる。

【0029】望ましい実施例では、各登録エントリは2つの項目、即ち、アップデータ・パス及びアップデータ・ネットワーク・アドレスを含む。そのパスは、アップデータ・コンポーネントが、ブート・アップ・プロセス中、オペレーティング・システムによって立ち上げられるように、アップデータ・コンポーネント・バイナリ・ファイルのロケーションである。これは、アップデータ・コンポーネントがいつもアクティブであり、作業を遂行するか又は他のアップデータ・コンポーネントからそれに発生されたリクエストを処理する準備ができていることを保証する。ネットワーク・アドレスは、ネットワークにおける他のコンピュータ・システム上のコンポーネントによって、ネットワーク上のそれを見つけるために及びそれとコミュニケーションするために使用されるアドレスである。

【0030】UNIX（商標）オペレーティング・システム及びTCP/IPプロトコル・スイートを使用するそのような登録の例は、アップデータ・コンポーネントに対する次のようなネーミング規約

SoftwareVendorName+_product_name+_updater
を使用する。

【0031】パス登録は、パス・エントリを記憶するためにUNIX/etc/inittabファイルに入れられる。例えば、IBM社のDB2（商標）データベース製品に対するアップデータ・コンポーネントが導入される時、それは次のようなフォーム、即ち、

ibm_db2pe_updater:2:respawn:/usr/sbin/db2_updater_binary

という/etc/inittabファイルにエントリを加えるであろう。

【0032】コンピュータ・システムがリブートする度に、それはこのファイルを読み取り、DB2アップデータ・コンポーネントを立ち上げるであろう。アップデータ・エントリにおける「respawn」キーワードは、アップデータ・コンポーネント・プロセスが一般的なシステム・オペレーション中に何らかの理由で障害を起こした場合、それはオペレーティング・システムによって自動的に再始動されることを保証する。この方法は、すべての導入されたアプリケーションに対するすべ

てのアップデータ・コンポーネントがいつもアクティブであることを保証するであろう。

【0033】ネットワーク・ロケーション登録は、UNIX/etc/serviceファイルに入れられる。例えば、DB2アップデータ・コンポーネントが導入される時、それは次のようなフォーム、即ち、

```
ibm_db2pe_updater 5000/tcp #net location of DB2
updater component
```

という/etc/servicesファイルにエントリを加えるであろう。

【0034】別のアップデータ・コンポーネントがDB2アップデータ・コンポーネントとコミュニケーションすることを望む時、それは、DB2アップデータ・コンポーネント名 ibm_db2pe_updater に関してこのファイルを検査することによってそれを見つけるであろう（実際には、UNIXコール getservbyname() によって間接的に行われる。その名前は標準的なネーミング規約に従って発呼者によって形成される）。それが見つかる時、それはDB2アップデータがポート番号5000における接続に関して聴取していることを知り、TCPプロトコルを使用するであろう。これは、問題のアップデータ・コンポーネントがDB2アップデータ・コンポーネントへのリンクを確立して会話（これについては後述する）を開始することを可能にする。

【0035】アップデータ・コンポーネントが他の遠隔の機械における他のアップデータ・コンポーネントを見つけてそれと通話するためには、上記情報は、両方の機械からアクセス可能であり且つそれを必要とするすべてのアップデータ・コンポーネントにとって利用可能なりポジットリ40'（望ましくは、ウェブ・ページ又はパンネットワーク・ファイルのようなそのネットワークにおける何処からもアクセス可能な中央データベース又は分散データベース）を持つことによって拡張されなければならないであろう。エントリは、updater_name machine_ip_address（又は、DNAエントリ）、ポート番号、プロトコルという形式のものであろう。

【0036】例えば、或る機構の製造部門は、分散ソフトウェア製品が相互に協同する3つのコンピュータ・システムを有することがある。それらのシステムはa、b、及びcと呼ばれる。ウェブ・ページ又はファイル manufacturing_collaborators.html における代表的なエントリは次のようになるであろう。

```
ibm_catia_updater    a.manufacturing.com    5000 t
cp
ibm_db2pe_updater    b.manufacturing.com    5100 t
cp
ibm_cics_updater     c.manufacturing.com    4780 t
cp
```

【0037】次に、アップデータ・コンポーネントは、IPアドレスを作成するためのDNA名と、遠隔のアッ

プデータ・コンポーネントがそのアドレスにおいて聴取しようとしているポート番号とを使用して他の任意のアップデータ・コンポーネントを接続し、それと通話することができる。

【0038】従って、導入時のアップデータ登録のステップは次のようになる。

(1) /etc/inittab ファイルにおけるエントリを作成する（アップデータ・プロセス・コード・ロケーションを登録する）

10 (2) /etc/service ファイルにおけるエントリを作成する（アップデータ・プロセス・ローカル・アドレスを登録する）

(3) 中央データベース・ファイルにおけるエントリを作成する（アップデータ・プロセス・パンネットワーク・アドレスを登録する）

【0039】導入プロセスはアップデータ・コンポーネントにウェブ・プロキシ・サーバのローカルIPアドレスを供給することも伴うことがある。多くの代替の登録実施方法が可能であることは当業者には明らかであろう。

20 【0040】アップデータ・コンポーネントは、それらに関連するソフトウェア製品に対する識別子及びバージョン番号のためのデータ・フィールドを含む。アップデータ・コンポーネントは、ヌル値にセットされたこれらのフィールドと共に顧客に配布可能であり、従って、導入手順は、アップデータ・コンポーネントが、識別子、現在のプログラム・バージョン、及びリリース番号を得るためにそのソフトウェア製品に質問するという初期ステップを含む。別の方法として、ソフトウェア・ベンダは、アップデータ・コンポーネント内に関連の製品ID及びバージョン番号を事前コード化することも可能である。

30 【0041】図1のシステム10は、複数の遠隔サーバ・システム50、50'を含むコンピュータのネットワーク100において接続されたものが示される。ローカル・システム10上に導入されたプログラムに更新を施すためのソフトウェア・リソースが遠隔サーバ・システム50、50'から得られる。各サーバ・システムは、そのサーバから得られるソフトウェア製品の最新バージョンのリスト60及びそのソフトウェア製品に対するパッチを記憶装置内に含む。各ベンダは、アップデータ・コンポーネントによって読み取り可能なフォーマットで製品リリース・履歴を含むソフトウェア更新のリスト60（その例が図2に示される）のようなウェブ・サイトを利用するものと仮定し、しかも所与のレベルから新しいレベルへのリリース（ソフトウェア製品リリースから新しいレベルへのこの遷移は以下では「成長パス」と呼ばれる）を形成するために必要なソフトウェア・リソース70を利用するものと仮定する。ソフトウェア更新リスト60におけるエントリは、各ソフトウェア製品

バージョン110に対して、その更新を施すために必要なソフトウェア・リソースの識別情報120及びその前提ソフトウェア製品及びそれらのバージョン番号の識別情報130を含む。或る場合には、必要なリソースはソフトウェア及び関連の導入命令の完全な置換バージョンである。別の場合では、リソースは、（例えば、エラー訂正のために）既存のプログラムを修正するためのパッチ・コード及びパッチの導入命令を含む。

【0042】現在の例に関して、ネットワーク100はインターネットであると仮定することにするが、本発明は任意のコンピュータ・ネットワークにおいて実施可能である。ネットワーク100にはサーバ・システム80が示され、そのサーバ・システムには、ネットワーク上の更新ソース・ロケーションを見つける場合に使用するためのサーチ・エンジン90が導入される。これは、ローカル・システム10から離れて設置されるように示されているが、必ずしもその必要はない。図面では、各アップデート・コンポーネント20は単一のプログラム30と関連付けて示され、しかも、すべての導入されたソフトウェア製品がそれらを管理する関連のアップデート・コンポーネントを有することは本発明のこの実施例の1つの特徴であるが、後述するように、これらの特徴のうちのどれも本発明にとって本質的なことではない。

【0043】次に、図3、図4及び図5を参照して、アップデート・コンポーネントのオペレーションを説明することにする。導入されたアップデート・コンポーネントが実行される時、その第1アクションは、導入時に得られた製品識別子及び製品バージョン・リリース番号をサーチ引数として1つ又は複数のサーチ・エンジン90に供給し、特定のソフトウェア製品に対して利用可能な更新に関するサーチを開始することである（ステップ200）。ソフトウェア・ベンダは、製品識別子及びリリース番号110によって参照される利用可能な製品更新のリスト60をそれらベンダのウェブ・サイトを介して供給するものと仮定すると、そのサーチは、更新情報が得られる関連のウェブ・サイト140を識別するであろう。サーチ・エンジンを始動しようとする初期の試みが不成功である場合、アップデート・コンポーネントは異なるサーチ・エンジン（第1のサーチ・エンジンに対して異なる地理的ロケーションにあってもよい）を始動するように試みるであろうが、別の方法として、事前設定された期間を待ち、しかる後、再試行してもよい。更新情報に対する関連のウェブ・サイト140を識別するURLはサーチの結果としてアップデート・コンポーネントに戻される（ステップ210）。

【0044】アップデート・コンポーネントはそのURLを使用してリスト60をアクセスし（ステップ220）、特定の製品に関連する利用可能な更新のリスト60の部分を含むファイル160をダウンロードする（ステップ230）。しかる後、アップデート・コンポーネ

ントは、図4及び図5に示されるステップ240-280を遂行する。各ファイル160は、デジタル的にサインされたメッセージ・ダイジェスト（例えば、MD5）を含む。次に、検索されたファイル160が、デジタル・シグニチャ・チェック・アルゴリズム（米国特許第5,231,668号に開示されたアルゴリズムのような）を使用して分析される（ステップ240）。これは、ファイル160が特定のソフトウェア製品に対する正しいソフトウェア更新リストを表すこと、及びそのファイルがサイン以後修正されていないことを確認するためには重要である。更に、デジタル・シグニチャのチェックは、これらが正しいもの以外に複数のウェブ・ページURLを含み得るので、サーチの結果を濾す有用な方法である（サーチは、ソフトウェア・ベンダによって発行されなかったページを含む、名前を付けられた製品バージョンに対する参照符合を持った他のページを見つけることがある）。ファイルをダウンロードし、それを検証しようとする試みが成功しなかった場合、アップデート・コンポーネントはそのサーチにおいて見つかった次のURLに進む。

【0045】次に、アップデート・コンポーネントは、ローカル・コンピュータ・システムにおいて、現在導入されているソフトウェア製品の識別子及びリリース番号と検索されたファイル160におけるリストされた利用可能な更新との間で比較を行う（ステップ250）。この比較は、現在のバージョンから更新バージョンへの可能な成長パスを決定するが、これらの可能な成長パスは、その後、事前定義された更新基準と比較され（ステップ260）、更新基準を満たさない如何なる可能な成長パスも廃棄される。従って、アップデート・コンポーネントは、現在のソフトウェア製品から利用可能な新しいバージョンに移行することが可能であるかどうか、及び現在同意されているライセンス条件の下でパッチを現在のバージョンに適用することが可能であるかどうかを決定する。

【0046】例えば、ソフトウェア製品のライセンスは、その製品に関する将来の如何なるバージョンへの移行及び利用可能な如何なるパッチの適用を可能にし、又は指定されたバージョンへの移行だけを可能にすることがあり、或いは、現在のバージョンにおけるエラーを修正又は訂正する利用可能なパッチの適用を許容するだけであることもある。現在のライセンスの制限のために利用し得ない可能な更新パスは、現在導入されているバージョンのソフトウェア・アセット・マネージャ（エンド・ユーザ又はIT調達マネージャでもよい）に送られたシステム発生メッセージとして通知され（ステップ270）、現在のライセンスが十分なものであるかどうかに関してそれらが判断することを可能にする。

【0047】起こり得る更新に関するライセンス制限の他に、アップデート・コンポーネントの更新基準又は成

長ポリシは、サイクル期間（例えば、1週間又は1ヶ月間）と、複数の可能な成長パスのうちのどれを選択すべきか（ライセンスによって許容される最新のバージョンをいつも選択するようにするか、又は最新のパッチをいつも選択し且つ新しいバージョンの利用可能性を通知するだけであるようにするか、或いは、前提ソフトウェアが既にローカル・システムにおいて得られる場合には新しいバージョンを選択するだけであるようにするか）を決定するための基準とを含む。成長基準は、アップデータ・コンポーネントによってダウンロードされる新しいバージョンにいつアップグレードすべきかというような制御情報も含むことがある。新しいソフトウェア製品バージョンに移行する時にデータ・マイグレーションが要求される場合、これが業務時間外で、或いは毎月又は毎年単一のスケジュールされた時間においてのみ行われることが必須であることがあり、これはアップデータ・コンポーネントによって制御可能である。

【0048】成長ポリシの定義は、現製品との同期を維持する必要がある時には前提ソフトウェア製品の更新がリクエストされるべきであるということを決するパラメータを含むこともある。これについては、更に詳しく後述することにする。アップデータ・コンポーネントによって設定及び適用される基準には大きな融通性があることは当業者には明らかであろう。

【0049】次に、アップデータ・コンポーネントは、更新基準を使用して可能な成長パスのセットから特定の成長パス（即ち、アップグレードするために利用可能なバージョン）を決定する（ステップ280）。例えば、アップデータ・コンポーネントは、更新基準によって許容される利用可能な更新のうちの最高の可能なバージョン又はリリース番号を、それが更新ポリシである場合、選択してもよい。

【0050】アップデータ・コンポーネントは、必要なソフトウェア・リソースが既にローカル・コンピュータ・システムにおいて利用可能であるかどうかをチェックするためにオペレーティング・システムのファイル・システムのスキャンを遂行する（ステップ290）（図3、図4及び図5参照）。その必要なリソースは、現在のアプリケーション・ソフトウェアを新しいレベルにもたすために必要なソフトウェア更新加工物であり、前提ソフトウェアを必要なレベルに更新することを必要とするソフトウェア更新である。前提導入済み製品と関連した各アップデータ・コンポーネントは、（a）それが導入されること、及び（b）それが、必要な前提レベルにあるか或いはそれよりも高いレベルにあること、を保証するために接触される（ステップ300）。すべての必要なリソースがローカルで又は他の機械において得られ且つ確認された場合、アップデータ・コンポーネントは、更新されたソフトウェア・バージョンを形成するステップ310（図5参照）に進む。それが確認されない

場合、アップデータ・コンポーネントは必要なリソースを得なければならない。

【0051】図3、図4及び図5に示されるように、更新されたバージョンを形成するための必要なソフトウェア・リソースがローカル・システムにおいて見つからなかった場合、アップデータ・コンポーネントは、その必要なリソースを見つけるために1つ又は複数のサーチ・エンジンに更なるリクエストを供給する（ステップ320）。サーチ・エンジンは1つ又は複数のURLを返送し（ステップ330）、アップデータ・コンポーネントはこれらを使用してソフトウェア・リソースを検索してローカル・コンピュータ・システムの記憶装置に入れる（ステップ340、350）。この段階では、アップデータ・コンポーネント又はユーザは、新しいバージョンにどのような訂正又は機能強化が含まれるかということに関する知識を全く持つ必要がない。ユーザがすべての更新の内容を調べるといふ労力を免れるように、更新基準は、どのタイプの更新が必要であるかを決定する。実際には、ユーザが更新の影響を決定することができることが望ましく、従って、更新のためのソフトウェア・リソースは、ユーザ又はアドミニストレータが読むことができるこれらの影響に関する記述を含む。

【0052】例えば、更新されるべきソフトウェア製品はワード・プロセッサ・アプリケーション・プログラムであってもよい。販売されたワード・プロセッサが或るフォントを脱落していたり或いはシソーラスを含まなかったりした場合、その後、これらのフィーチャを加えるためのパッチが利用可能にされることがある。アップデータ・コンポーネントは、更新基準次第でこれらをそのワード・プロセッサに加える機能を有する。

【0053】本発明の別の実施例では、更新リストに関する初期サーチに続く必要なソフトウェア・リソースに関するサーチは不必要である（或いは、前提ソフトウェア製品及びパッチ又は現製品に対する新しいバージョンがある場合に必要であるだけである。下記参照）。これは、現製品によって直接に必要とされる更新ソフトウェア・リソースが必要なリソースのリストと関連して記憶されるためである。即ち、そのリストは、そのリストからの成長パスの選択が必要な更新のネットワーク・ロケーションに対するポインタ（場合によっては、更に、前提ソフトウェア製品のロケーションに対するポインタ）の選択を伴うように、その必要なリソースのネットワーク・ロケーションに対するポインタを含む。

【0054】デジタル・シグニチャ・チェックによる第2の検証が、今度は、ダウンロードされたリソースに関して遂行される（ステップ360）（図5参照）。ダウンロードされたリソースの正当性を検証した後（ステップ360）、アップデータ・コンポーネントは、更新ポリシに従ってターゲット環境における導入を自動的に行う（ステップ310）。実際には、これは、管理パス

ワードのようなユーザからの情報又はデータベース使用パラメータ値を必要とすることがあるが、本発明の望ましい実施例では、ダウンロードされたコードの導入は、ユーザが何らかの導入情報を余所から知る又は余所から得ることをそれが必要としないという意味で、及び、アップデータ・コンポーネントが自動的に更新を施すことを事前定義の更新基準が可能にする場合、一般に、ユーザが実行時に如何なる判断も行わなくてよいことをそれが可能にするという意味で自動的である。

【0055】シェルにおいてエンコードされた機械読み取り可能な導入命令を（例えば、Script、又はPERLのような解釈言語、又はマイクロソフト社のWindows（商標）オペレーティング・システム上のアプリケーションの場合のsetup.exeのような実行可能なものとして）含むことはよく知られている。本発明によるアップデータ・コンポーネントは、機械読み取り可能な命令を関連のソフトウェア・リソースと共にダウンロードし（ステップ350）、それらを自動的に実行するであろう（ステップ310）。従って、アップデータ・コンポーネントは導入命令を自動的に処理し、従来方法では必要とされた人からの入力を回避する。Scriptは、アップデータ・コンポーネントの第1バージョンを導入した第1の導入者から集められた情報（例えば、ユーザ名及びアプリケーション・アドミニストレータのパスワードのような情報、及び導入ディレクトリ等）を再使用するように適応可能である。

【0056】本発明の望ましい実施例による更新の方法は、ソフトウェア・ベンダが、形成する必要のあるソフトウェア・リソースを1つの製品レベルから他の製品レベルに編成することを必要とする。例えば、バージョン1.1.1からバージョン1.1.4への移動は、一般には、施されるべき一連のパッチ、及び機械処理可能な導入命令にうまくエンコード可能である場合の必要な導入順序を含むであろう。そこで、ユーザは、修復及び機能強化を施す順序をユーザが制御することを必要とする方法では固有である努力及び人的エラーの危険を免れる。従って、1つの製品レベルから別の製品レベルに如何に移行すべきかという問題は、顧客に代わってソフトウェア・ベンダによって処理され、アップデータ・コンポーネントは、ベンダによってサポートされるレベル（即ち、特定の既存の製品レベルに対してソフトウェア・ベンダにより発行された成長パス）までしか移ることができない。

【0057】アップデータ・コンポーネントはレポートを発生し（ステップ380）、ログ・レポートに書き込み（ステップ390）、しかる後、所定の更新サイクル期間（反復期間パラメータは、アップデータ・コンポーネント導入された時に形成される）の満了時に再び活性化される（ステップ410）まで実行を終了する（ステップ400）（望ましい実施例では、アップデータがス

リープ又はアイドル状態に進む）。

【0058】A1. アップデータ・コンポーネントの構造

アップデータ・コンポーネントの構造は、データ、そのデータを操作するための方法、及び他のアップデータ・コンポーネントがそれと接触及びコミュニケーションすることを可能にする汎用アプリケーション・プログラミング・インターフェースより成る。次に、この構造を詳しく説明することにする。

10 【0059】アップデータ・コンポーネント・データ：アップデータ・コンポーネントはつぎのような永続的データを含む：

Product_ID： このアップデータ・コンポーネントによって管理されるソフトウェア製品の識別子。

Current_Installed_Version： 導入されたソフトウェアに対するバージョン識別子（例えば、バージョン3.1.0）。

20 Current_License： 現在のソフトウェア・ライセンスによってユーザがアップグレードし得るソフトウェア製品バージョンに対応するバージョン識別子（例えば、4.0.z）。別の方法として、これは、機械読み取り可能なライセンス条件をアクセスする時に使用するためのライセンス識別子（例えば、LIC1）であってもよい。

30 Installation_Environment： 属性名／属性値の対のリスト。これは、アップデータ・コンポーネントが初めて使用された時、ユーザによって入れられた値を記憶するためにアップデータ・コンポーネントによって使用される。例えば、アップデータ導入ユーザID及びパスワード、或いは、ルート・パスワード、導入ディレクトリ、ウェブ・プロキシ・サーバ・アドレス、サーチ・エンジンURL、ログ・ファイル名、ソフトウェア・アセット・マネージャ電子メール・アドレス等。このデータは、その後の自動更新が必要とされる時に再使用されるであろう。

成長ポリシ・パラメータ：

40 a. Growth_Cycle： アップデータ・コンポーネントが毎日、毎週、又は毎月そのソフトウェア製品を更新しようと試みなければならないかどうかを決定するデータ。

b. Growth_Type： 更新がバグ修復及び機能強化（即ち、パッチ）のみに制限されるか、或いは、各成長サイクルにおける最新のリリースへのアップグレードを必要とするかを決定するデータ。

50 c. Force_Growth： （イエス／ノー）アップグレードするように他のソフトウェア・リソースを強制すべきかどうかを、それがこのソフトウェアのアップグレードのための前提である場合に決定するパラメータ。（或る実施方法は、アップグレードするように他のソフトウェアを強制することによって、この単純なイエス／ノーより

ももっと融通性のある制御を提供するであろう)。

Last_Growth_Time : アップデータ・コンポーネントが最後に実行された時の日付及び時間。

【0060】アップデータ・コンポーネントは次のような非永続的データも含む：

Possible_Growth_Paths : 利用可能なアップグレード・パスを表す一時データ (例えば、バージョン番号 3.1.d、3.2.e、4.0.a)。

【0061】A2. 私的アップデータ機能：アップデータ・コンポーネント・ロジックは次のような方法を含む：

Discover_Possible_Growth_Paths() :

インターネット (又は、他のネットワーク) におけるこのソフトウェアのための **Growth_Path** 情報に関するサーチ。このサーチ方法は、標準的なサーチ・エンジン・サーバを介してサーチを開始する。戻された情報は更に新しいバージョン及び関連の前提製品情報である。次に、**Growth_Path** 情報は成長ポリシパラメータに従って減少する。**Growth_Path** リストにおけるすべてのメンバに対して、前提製品の適正なバージョンがローカル・コンピュータ及び／又はリモート・コンピュータにおいて得られるかどうかに関するチェックが行われる。これらの前提製品を管理するアップデータ・コンポーネントはアクセスされ、これがポリシである場合、成長することを強制される。すべての事前製品が正しいレベルでローカルに存在する場合、又はネットワーク上で遠く離れて得られ、しかも、「強制成長」ポリシと共に存在する場合、ソフトウェア製品のより新しいバージョンに対する識別子が **Possible_Growth_Paths** リストに加えられる。

Decide_Growth_Path() :

成長ポリシを解釈し、単一の成長パスを選択する。本発明の或る実施方法は、例えば、他のプログラムへの更新を強制すべきかどうかというような考察事項が存在する場合、ユーザ対話がそのパスを選択することを伴うであろう。

Get_Resources(Parameter: Chosen_Growth_Path) :

Chosen_Growth_Path (例えば、3.2.0) を与えられた場合、必要なリソース (パラメータ **Product_ID**、**Current_Installed_Version**、**Chosen_Growth_Path**) に関してサーチし、すべてのリソースをローカル・コンピュータにダウンロードする。これは、新しいバージョン及び機械処理可能な導入命令を必要とするソフトウェアを含むであろう。

Install_Resources() :

必要なファイルを正しいロケーションに導入すること、或いは、それらのファイルをコンパイルすること、及びソフトウェアを収容するために既存のシステムの構成を修正することを含み、すべてのアクションをファイルにログする (及び「アンインストール」方法がすべてのア

クションをやり直すことを可能にする) 導入命令を処理する。

Grow() :

方法を開始する：

Discover_Possible_Growth_Paths()

可能な成長パスが存在しない場合、アップデータ・コンポーネントはアイドル状態になるさもなければ、

Decide_Growth_Path()

Get_Resources(Parameter: Chosen_Growth_Path)

10 **Install_Resources()**。

Grow() は、すべての完了したアクションをログに書き込み、アップデータ・コンポーネントの実行を終了する。アップデータ・コンポーネントは、新しい更新要件に関して再びチェックすべき時間まで、又はそうするように他のアップデータ・コンポーネントによって指示されるまでアイドル状態になる。

【0062】A3. 汎用アップデータ・コンポーネント API

アップデータ・コンポーネントは下記のような汎用 API を含む。これらの機能は、リモート・プロシージャ・コール、メッセージ指向ミドルウェア、ORB (オブジェクト・リクエスト・ブローカ) 等のような既存のネットワーク通信ソフトウェアを使用して呼び出し可能であろう。

Get_Release() :

この機能は他のアップデータ・コンポーネントによって呼び出され、そしてこのアップデータ・コンポーネントによって管理される製品のリリース・レベルを戻す。

Update(new_level) :

30 他のアップデータ・コンポーネントがこの機能を呼び出し、このアップデータ・コンポーネントによって管理される製品を、**new_level** パラメータ値によって表された新しいレベルに移す。これは私的機能 **Grow()** を呼び出す。

Receive_Event(event details) :

アップデータ・コンポーネントがアップグレードするようにリクエストを受ける時、それは、それがいつ更新を完了したかを呼出のアップデータ・コンポーネントに知らせなければならない。他のアップデータ・コンポーネントに代わって更新を遂行するアップデータ・コンポーネントは、その更新の成功を伝えるためにリクエストしたアップデータ・コンポーネントのこの機能を呼び出すであろう。イベント詳細は「製品 ID、新リリース・レベル、ok」又は「製品 ID、新リリース・レベル、失敗」のようなストリングであってもよい。

【0063】更新の強制を可能にすること (又は、更新の強制が更新ポリシの一部でない場合、ソフトウェア・アセット・マネージャに通知を送ること) による非同期化された前提製品の潜在的問題点の自動処理は、従来技術の更新方法よりも優れている重要な利点である。

【0064】初期サーチに応答してアップデータ・コンポーネントに戻された更新リスト・ファイル160は前提ソフトウェアの識別子130を含むので、その情報は、前提ソフトウェアがローカルで入手可能に又はリモートで入手可能かを、アップデータ・コンポーネント登録データベース40、40'の前述の検査(ステップ290)がチェックすることを可能にする。それがローカルで設置された又はリモートで設置されたアップデータ・コンポーネントをすべて見つける場合、前提ソフトウェアが得られることは確かであり、次に、すべての前提が正しいレベルにあることを確実にするために各ソフトウェア製品に対する各アップデータ・コンポーネントと接触することが必要である。前提ソフトウェア30'に対する必要な製品識別子を有するが、必要なバージョン番号を持たないアップデータ・コンポーネント20'がローカルで又はリモートで見つかった場合、及び更新の強制が更新ポリシーである場合、第1コンピュータ・プログラムのアップデータ・コンポーネント20はこの前提アップデータ・コンポーネント20'と接触し(ステップ300)、それがその関連の前提ソフトウェア製品30'の更新を試みることをリクエストする。このアップデータ・コンポーネント20'は、必要な場合には、その前提ソフトウェアの他のアップデータ・コンポーネントにそれらのバージョンを更新するようにリクエストする。

【0065】或る段階において、関連のアップデータ・コンポーネントがローカルで又はリモートで見つからない場合、関連製品を更に成長させるためには、新しい製品に対する要件を知らせるためにアセット・マネージャに送られる。新しいレベルに成長させるためのアップデータ・リクエストの連鎖時の或る段階で、1つのアップデータ・コンポーネントが必要なレベルに移ることができなかった場合、この失敗はそれが呼び出したアップデータ・コンポーネントに報告され、そのアップデータ・コンポーネントはそのコンポーネント更新オペレーションの失敗等を、トランザクション全体を開始させたアップデータ・コンポーネントにプロンプト指示する。

【0066】従って、それらの更新基準によって定義されたそれらの自動的な行為の他に、アップデータ・コンポーネントは他のアップデータ・コンポーネントからのリクエストのように外部刺激に反応することができる。

【0067】B. 更新同期化の例

次に、2つの製品の間の更新同期化の実施方法の例を説明することにする。この例は、すべての製品が存在し、互換性のあるリリース・レベルにあるように、一方のアップデータ・コンポーネントが前提ソフトウェアを同期化するように他方のアップデータ・コンポーネントとコミュニケーションする方法を示す。

【0068】CORBA(共通オブジェクト・リクエスト・ブローカ・アーキテクチャ)ORB(オブジェクト

・リクエスト・ブローカ)は2つのアップデータ・コンポーネントの間のロケーション及びコミュニケーションのために使用される。上記汎用APIを使用すると、或るアップデータ・コンポーネントがネットワーク上の他のアップデータ・コンポーネントと会話することができるように、CORBAプログラミングの技術に詳しい人がコミュニケーション・コードを開発することは簡単なことである。この例では、アップデータ・コンポーネント登録データベース40は、各導入されたアップデータ・コンポーネントに対して「updater_component_name.iop」と呼ばれるファイルを含む、ネットワークを介して得られるディレクトリ又はフォルダである(なお、iopは、インターオペラブル・オブジェクト・リファレンスを表す)。

【0069】このファイルは、例えば、CORBA機能、即ち、

CORBA::Object::_string_to_object() in C++

を使用してファイルを読み取る任意のアップデータ・コンポーネントによってそのアップデータ・コンポーネントに対する基準に変換され得る一連のバイトを含む。

【0070】更に、この基準は、対応するアップデータ・コンポーネントに対する独特のアドレスをそれが表すので、ネットワーク上のどこにあるアップデータ・コンポーネントに対しても基準になり得る。アップデータ・コンポーネントAがアップデータ・コンポーネントBに対する基準を作った時、アップデータ・コンポーネントAは、例えば、C++ mapping A->Get_Release()を使用することによって汎用API関数を呼び出すことができる。それは、その後、アップデータ・コンポーネントAによって管理されるソフトウェアのリリース・レベルの値に戻すであろう。

【0071】この例では、2つの製品、即ち、それぞれ異なる機械M及びNにおけるIBM社のDB2製品及び「照会ビルダ」と呼ばれる照会ツールを考察することにする。(機械M及びNは同じ機械であってもよい;この例は単に、それらが別個であってもよいことを示す)。両方の製品とも、簡単に概説したように、CORBA ORBアーキテクチャを使用するアップデータ・コンポーネントを有する。ORBコミュニケーション・デーモンは参加システムM及びNにおいてアクティブである。

【0072】ステップ1. 登録フェーズ: DB2アップデータ・コンポーネントは、オペレーティング・システムがシステムMにおいて始動し、ネットワーク・ファイル・システム・フォルダ又はディレクトリにおけるibm_db2_updater.iopと呼ばれるファイル(そのファイルに関するその後のサーチを助けるために使用される或るネーミング標準による)を直ちに作成する。このディレクトリは必ずしもM又はNではない任意の機械においてホストにされる。そのファイルは、アップデータ・コンポーネントに対する基準を作るために使用可能な一連

のバイトを含む。

【0073】 [擬似コード]

```
FileHandle=open("/network/filesystem/directory","i
bm_db2_updater.iop");
ReferenceBytes=CORBA::Object::_object_to_string();
Write(FileHandle, ReferenceBytes);
close(FileHandle);
```

【0074】 QueryBuilder アップデータ・コンポーネントが始動してその登録を同じディレクトリ又はフォルダに書き込み、この場合には、再び、ファイル `ibm_querybuilder.iop` を呼び出す。

【0075】 この段階では、両方のアップデータ・コンポーネントがアクティブであり、それらの存在及びロケ

[擬似コード]

```
if (dbref =
CORBA::Object::_string_to_object(readfile(ibm_db2_updater.iop)))
then SUCCESS: 我々はアップデータに接続された
else
    FAIL: 前提ソフトウェアはコラボレート・シ
    ステムのセットには存在しない。ソフトウエ
    ア・アセット・マネージャに電子メールを送
    り、状況を通知する。
    新バージョンに成長する試みを中止する。
```

endif

【0079】 ステップ3. この段階で、我々はネットワーク化されたコンピュータのセットにおけるどこかにDB2が存在することを知る。今や、我々は、それが正しいレベルにあるかどうかを知る必要がある。我々は、QBアップデータ内から上記の汎用API機能 `Get_Release()` を実行することによってこれは簡単に行う。従って、QBアップデータは、それに関して何かを行うように、即ち、それがどのようなリリースであるかを知らせるようにDB2アップデータにリクエストするクライアントである。

【0080】 [擬似コード]

```
db2_release = dbref->Get_Release();
```

【0081】 例えば、これは値「2.0」を戻す。

[擬似コード]

```
dbref->Update("2.1", QBref); // QBref はQBアップ
// データに対する既製の基準である。DB
// 2アップデータがそれ自身を更新する試
// みを終了した時、それが直ぐに成功又は
// 失敗というその結果を送ることができる
// ように、DB2アップデータに送られる
```

```
EVENT= null
```

```
While (EVENT equals null)
```

```
{何もしない;}
```

```
if (EVENT equals "SUCCESS")
```

```
then このプロセッサ・コンポーネント (即、Query
Builder) によって管理されるソフトウェアを
```

*ーションをネットワーク・ディレクトリに登録している。

【0076】 ステップ2. QueryBuilder はバージョン1からバージョン2に成長しようとするが、前提はDB2バージョン2.1又はそれ以上である。下記のアクション・シーケンスが生じるであろう。QueryBuilder はQBとして表され、DB2はDB2として表される。

【0077】 QB: ネットワーク・ディレクトリにおいてファイル `ibm_db2_updater.iop` (標準に従って作られたファイル名) に関してサーチする。それはファイルを見つけ、それを読み取り、それを使用可能な基準に変換する。

【0078】

【0082】 ステップ4

クライアント側: QBアップデータ・コンポーネントは、これが十分ではないことを知っており、それはバージョン2.1を必要とする。それはその `Force_Growth` パラメータを調べる。そのパラメータにおいて、例えば、「イエス」は、前提ソフトウェアを、それがそれ自身の更新プロシーダを遂行する前に必要なレベルまで成長させなければならないことを意味する。従って、QBアップデータは、新しいリリースまで成長するようにDB2アップデータに通知し、しかる後、前提ソフトウェアが新しいリリースに成長するまで、又がそうすることに失敗するまで待つ。

【0083】

成長させようとする

else

失敗をログに書き込む;

成長させようとしない;

スリープに進み、その後トライする;

endif.

【0084】サーバ側：DB2アップデータ・コンポーネントは成長するというリクエストを受ける。それは成長するように試みる。

【0085】それは結果を呼出クライアントに報告する * 10

[擬似コード]

DB2 attempts to grow.

if Growth Successful then

QBRef->Receive_Event("SUCCESS"); // 機能 Receive_

// Event の実施はQBアップデータ・コンポ

// ーネントにおける EVENT と呼ばれる変数を

// APIコールにおいて送られたパラメータ

// の値、即ち、IFステートメントのこのセ

// クション内にある場合には "SUCCESS" に単

// にセットすることに注意して欲しい。

else

QBREF->Receive_Event("FAILURE");

end if

【0087】前述のように、事前定義された更新基準は、利用可能な更新セットのうちのどれが適用されるべきか、及びどれが無視されるべきかを決定することができる。更新基準は、ソフトウェア更新が利用可能であるとして識別されるがこの更新の適用が更新ポリシーではなく或いは不可能である時、エンド・ユーザ又はシステム・アドミニストレータに通知を送るためのアップデータ・コンポーネントへの命令を含むことができる。前に示された例の1つは、更新ポリシーが前提ソフトウェア製品のアップグレード又はデータのマイグレーションを必要とすることがあり（例えば、ソフトウェア製品がデータベース製品である場合）、一方、それが何らかのエラー訂正パッチを導入することを意図したポリシーであり得るので、その更新ポリシーがソフトウェア製品の完全置換バージョンを導入することではないということである。一方の製品をアップグレードすることが他方の前提相補製※

製品ID: テスト

現導入済みバージョン: 1.0.a

現ライセンス: LIC1

導入環境: "USERID:TestOwner, USERPASSWORD:easy"

"INSTALLPATH: /usr/bin/testapp"

成長サイクル: 週

成長タイプ: パッチ、最新、自動的

強制成長: なし

最終成長時: 08/10/97 月曜日

アップデータは毎週、例えば、各月曜日の夜の午前3時に実行される（タイミングを決定するのはシステム・ア

*（それは機能呼出において呼出元に対する基準を受けるので、呼出クライアントと接触する方法を知っている）。

【0086】

※品のアップグレードを必要とする場合、更新の自動導入よりもむしろ通知が実施可能である。

【0088】更新ポリシーは、アップデータがユーザ又はアドミニストレータからの入力をリクエストする環境を定義することによって、更新プロセスの自動化の程度も決定することができる。

30 【0089】次に、特定の例のアップデータ・コンポーネントの実行を、更に詳細に説明することにする。このアップデータ・コンポーネントの機能は、「テスト」と呼ばれる導入済みの製品を、すべてのリリースされたパッチを有する全体的に最新の状態に維持することであって、テストの置換バージョンを導入することではない。まず、アップデータ・コンポーネントが次のようなデータ・インスタンス化によって構成される。

【0090】

ドミニストレータである）。

50 【0091】以下には、この例のアップデータ・コンポ

ーメントに対する可能な実行トレースが示される。

【0092】実行トレースの例

ステップ1. 成長サイクルが開始する:

```
>>>> START : Discover_possible_Growth_Paths()
```

* フレーズ ("IBM Test 1.0.a Growth Paths") を使用してリモート・サーチ・エンジン (例えば、インターネット・サーチ・エンジン) におけるサーチを実行する。サーチは、ベンダが製品に対する現在の成長パスを概説することによって公表されたURLを戻す。

* URLをダウンロードする: ファイル内容は:

```
"1.0.b,none; 2.0, other_required product_product_id 1.0.c;"
```

* ハッシュ・アルゴリズム及びデジタル・シグニチャを使用してURLファイルを認証する。真正でない場合、他のURL適合基準に関するサーチに戻る。

* growth_path_list を形成する: growth_path list = "1.0.c,none;2.0,other_required_product_id 1.0.c;"

* すべてを除去するが、パッチ・レベルは Growth_path リスト (即ち、第1バージョン及び1.0にマッチした第2リリース番号を有するものだけ) から増加する (Growth_Policy による)。

* growth_path list = "1.0.b,none;"

* リストにおける全メンバに対して、前提が存在することを保証する。この例では、リストのすべてのメンバがこの基準に普通に合致する。

* 候補の growth_paths を Possible_Growth_Paths list = 1.0.b に入れる。

```
<<<< END : Discover_possible_Growth_Paths()
```

【0093】ステップ2. 次に、アップデータ・コンポーネントは成長パスを介して以下の過程をたどる:

```
>>>> START Decide_Growth_Path()
```

* 成長ポリシは、最新のパッチされた改訂に我々が成長しなければならないことを指令する。(この例では、最新の改訂は普通であること、即ち、それは1.0.bであることを決定する)

* chosen_growth_path = 1.0.b

```
<<<< END : Decide_Growth_Path()
```

【0094】ステップ3. 次に、アップデータ・コンポーネントは、現在のソフトウェア・レベルを新しいソフトウェア・レベルに修正するための必要なリソースを得る。

```
>>>> Get_Resources()
```

* フレーズ ("IBM Test REVISION 1.0.a to 1.0.b RESOURCES") を使用してリモート・サーチ・エンジン (例えば、インターネット・サーチ・エンジン) においてサーチを実行する。

* サーチはURL、例えば、次のものを戻す。

```
ftp://ftp.vendor-site/pub/test/resources/1.0.a-b"
```

* アップデータはURLによって指示されたファイルをダウンロードし、それが真偽を検査する機密保持エリア

に入れる。

* アップデータは真偽を (例えば、RSAアルゴリズムに基づくデジタル・シグニチャ、又は他の方法を使用して) 検査する。

ファイルが真正でない場合、サーチに戻る (下記の注意1参照)

* アップデータはリソースをアンパックして一時ディレクトリにする (下記の注意2参照)。これらのリソースは機械処理可能な導入命令 (例えば、UNIXシェルスクリプト又はMVS REXXのようなスクリプト言語で書かれた命令) 及び実際にはソフトウェア・フィックスを含むファイル (バイナリ・コンパイル又は要求コンパイル) を含む。

```
<<<< END : Get_Resources()
```

【0095】上記のタスクに関する注意

注意1 - 時間を節約するために、アップデータは「シグニチャ」と呼ばれ、URLを含む標準ファイルを、URLのダウンロードの前に捜す。

```
ftp://ftp.vendor-site/pub/test/resources/1.0.a-b
```

及びその内容のリスト。

これはハッシュされ、サインされる。このシグニチャを使用して、アップデータ・コンポーネントは、(或る範囲までの) URLの真偽を、そのダウンロードの前に迅速に確立し、最後のダウンロードされたリソースが一時ディレクトリにアンパックされた後にそれらのリソースを確認するためにその情報、即ち、ファイル・リストを使用することができる。最後のURLがダウンロードされる時、それは再び真偽をチェックされる (誰かが真正なURLロケーションに贗造物を配置しないように保護するために)。

注意2 - アンパッキングの部分は、アップデータ・コンポーネントが導入スクリプトを調べ、必要な場合には、その導入環境データの内容に基づいてそれらを修正することである。例えば、導入命令がシェルスクリプトでコード化される場合、それは INSTALLPATH のすべてのインスタンスをトークン "/usr/bin/testapp/" でもって置換するであろう。再び、属性のネーミング規約が、導入命令におけるトークン代用の方法であるので標準化される。これは、全体的に自動的導入を可能にする。

【0096】ステップ4 次に、アップデータ・コンポーネントは実際のソフトウェア・アップグレードを実施する。

```
>>>> START Install_Resources()
```

* 導入命令を実行する。

* 次のような値を更新する:

```
Current_Installed_Version = 1.0.b
```

```
Last_Growth_Time = Date+Time
```

* アップグレードが効果を現す前に、導入と、オペレーティング・システムのリブート又はアプリケーションの

再始動が必要であるかどうかとを知らせる電子メールをソフトウェア・アセット・マネージャに送る。

<<<< END Install_Resources()

【0097】これはこの現成長サイクルの終了である。シードは現時点で Last_Growth_Time 値を更新し、しかる後、終了する。このサイクルのために費やされる時間は、現在導入されているバージョンに対するアップグレード・パスがないことをアップデータ・コンポーネントが知っている場合の数秒から、現在のものからの全体的に新しいリリースがダウンロードされて新しい前提ソフトウェアと共に導入されることになる場合の数時間までのどれかになり得るであろう。

【0098】詳細に説明した上記実施例に対する代替は異なる各ソフトウェア製品に対して独立のアップデータ・コンポーネントを必要とせず、各製品と共にダウンロードされる製品特有のプラグ・イン・オブジェクト及び命令と共にシステム上に導入される単一の汎用アップデータ・コンポーネントを使用する。これらのオブジェクトは、汎用コードと相互協調して上記製品特有のアップデータ・コンポーネントの同じ機能を提供する。システム上に導入されたすべてのアプリケーション・プログラムではなく或るアプリケーション・プログラム及び他のソフトウェア製品が関連のアップデータ・コンポーネントを有するというようなシステムにおいて本発明が実施可能であること及び本発明の技術的範囲内で上記実施例に対する別の変更が可能であることは当業者には明らかであろう。

【0099】まとめとして、本発明の構成に関して以下の事項を開示する。

【0100】(1) コンピュータ・ネットワーク内で接続されたコンピュータ・システム上に導入された1つ又は複数のコンピュータ・プログラムを更新する場合に使用するためのアップデータ・コンポーネントにして、1つ又は複数の必要なソフトウェア・リソースを検索するために、前記1つ又は複数の必要なソフトウェア・リソースが設けられた前記ネットワーク内の1つ又は複数の識別可能なロケーションへのアクセスを開始するための手段と、1つ又は複数の検索されたソフトウェア・リソースを使用して前記導入されたコンピュータ・プログラムの1つにソフトウェア更新を施すための手段と、を含むアップデータ・コンポーネント。

(2) 使用可能な関連の更新リソースを識別するために、前記1つ又は複数の識別可能なロケーションから得られるソフトウェア更新リソースと前記コンピュータ・システム上に導入された1つ又は複数のコンピュータ・プログラムとの比較を行い、前記使用可能な関連の更新リソースと前記コンピュータ・システムにおいて記憶された事前定義の更新基準とを比較するための手段と、ソフトウェア・リソースを自動的にダウンロードし、前記事前定義の更新基準を満たすソフトウェア更新を施すた

め的手段と、を含む上記(1)に記載のアップデータ・コンポーネント。

(3) 前記事前定義の更新基準は適用可能なソフトウェア製品ユーザ・ライセンスによる許容し得る更新の範囲の定義を含む上記(2)に記載のアップデータ・コンポーネント。

(4) 前記ソフトウェア更新を施すための手段は前記事前定義の更新基準に従って及び更新のためにダウンロードされたソフトウェア・リソースの一部である導入のためのコンピュータ読み取り可能な命令に従って、使用可能な関連のソフトウェア・リソースを導入するための手段を含む上記(2)又は上記(3)に記載のアップデータ・コンポーネント。

(5) 1つ又は複数のロケーションを識別するための情報が前記アップデータ・コンポーネントによって保持され、コンピュータ・プログラム製品の製品識別子を含み、前記アップデータ・識別子は前記製品識別子をサーチ・エンジンに供給するように適応し、前記製品識別子は前記サーチ・エンジンがネットワーク・ロケーションを識別するために使用するためのサーチ・パラメータとして働く、上記(1)乃至上記(4)の何れかに記載のアップデータ・コンポーネント。

(6) 前記アップデータ・コンポーネントは、使用可能なソフトウェア更新リソースのリストが保持されているネットワーク・ロケーションを前記サーチ・エンジンが識別することに応答して前記リスト及び前記リソースの前提ソフトウェア製品をダウンロードし、前記リスト及び前提ソフトウェア製品と前記コンピュータ・システム上に導入されたコンピュータ・プログラムとを比較し、前記前提ソフトウェア製品に対する更新が必要である場合に前記前提ソフトウェア製品に対する更新をリクエストするように適応する上記(5)に記載のアップデータ・コンポーネント。

(7) 前記アップデータ・コンポーネントはコンピュータ・システム上に前記アップデータ・コンポーネントを導入するための機械読み取り可能な導入命令を有し、前記導入命令は前記アップデータ・コンポーネントが他のアップデータ・コンポーネントによって識別可能及び接触可能であるように他のアップデータ・コンポーネントによってアクセスし得るリポジトリによって前記アップデータ・コンポーネントを登録するための命令を含む、上記(1)乃至上記(6)の何れかに記載のアップデータ・コンポーネント。

(8) 前記アップデータ・コンポーネントは、現在のアップデータ・コンポーネントがそのコンピュータ・プログラムを更新することを相補的なコンピュータ・プログラムがリクエストする時に介するAPIを含み、前記現在のアップデータ・コンポーネントは更新リクエストに応答してそのコンピュータ・プログラムを更新するために更新方法を呼び出すように適応し、前記現在のア

ップデータ・コンポーネントは、そのコンピュータ・プログラムが前提コンピュータ・プログラムの更新を必要とする時、システム発生されたリクエストをそのコンピュータ・プログラムの前記前提コンピュータ・プログラムのアップデータ・コンポーネントに送るように適応する上記（６）又は上記（７）に記載のアップデータ・コンポーネント。

（９）前記更新を施すための手段は存在する導入済みのソフトウェアを修正する訂正及び機能拡張ソフトウェアを導入し、導入されたソフトウェアを置換する導入済みのソフトウェアのアップグレードしたバージョンを導入するように適応する上記（１）乃至上記（８）の何れかに記載のアップデータ・コンポーネント。

（１０）コンピュータ読み取り可能な記録媒体上に記録されたコンピュータ・プログラム・コードを含み、前記コンピュータ・プログラム・コードは前記コンピュータ・プログラム・コードを更新するための上記（１乃至上記（９）の何れかに）に記載された統合アップデータ・コンポーネントを含むコンピュータ・プログラム製品。

（１１）コンピュータ読み取り可能な記録媒体上のレコードのためのコンピュータ・プログラム・コードを含み、前記コンピュータ・プログラム・コードは前記コンピュータ・プログラム・コードを更新するための上記（１）乃至上記（９）の何れかに従って統合アップデータ・コンポーネントを含むコンピュータ・プログラム製品。

（１２）コンピュータ・ネットワーク内で接続されたコンピュータ・システム上に導入されたコンピュータ・プログラムを自動的に更新するための方法にして、前記コンピュータ・プログラムを更新する場合に使用するためのアップデータ・コンポーネントを前記コンピュータ・システムに配布するステップと、前記コンピュータ・プログラムを現在のバージョンから更新済みのバージョンに形成するためのダウンロード可能なソフトウェア・リソースを第１ネットワーク・ロケーションに提供するステップと、前記コンピュータ・システムにおいて実行される時、前記アップデータ・コンポーネントが遂行するように適応するステップであって、（ａ）前記アップデータ・コンポーネントによって保持された情報から識別可能であり、又は前記アップデータ・コンポーネントによってアクセス可能であって、前記ソフトウェア・リソ

ースが配置される前記第１ネットワーク・ロケーションへのアクセスを開始するステップと、（ｂ）前記ソフトウェア・リソースを前記コンピュータ・システム上にダウンロードするステップと、（ｃ）ダウンロードされたソフトウェア・リソースを使用して前記コンピュータ・プログラムを前記現在のバージョンから前記更新済みのバージョンに更新するステップと、を含む方法。

（１３）前記アップデータ・コンポーネントにおける情報から識別可能な第２ネットワーク・ロケーションに前記コンピュータ・プログラムにとって使用可能な更新のコンピュータ読み取り可能なリストを設けるステップを含み、前記アップデータ・コンポーネントによって、前記第１ネットワーク・ロケーションにアクセスする前に遂行されるように適応するステップにして、前記リストを検索するために前記第２ネットワーク・ロケーションへのアクセスを開始するステップと、使用可能な関連の更新リソースを識別するために、前記リストを読み取り、リストされた使用可能な更新と前記第１コンピュータ・システム上の前記コンピュータ・プログラムとの比較を行うステップと、前記使用可能な関連の更新リソースと前記アップデータ・コンポーネントにおける事前定義された更新基準とを比較し、前記更新基準を満たす更新のための使用可能な関連の更新リソースを識別するステップと、を含む上記（１２）に記載の方法。

【図面の簡単な説明】

【図１】導入されたアップデータ・コンポーネントを有するローカル・コンピュータ・システム、使用可能な更新のリスト及び更新を施すためのソフトウェア・リソースを記憶するサーバ・コンピュータ、及びサーバを見つけるためのサーチ・エンジンを含むコンピュータ・ネットワークの概略図である。

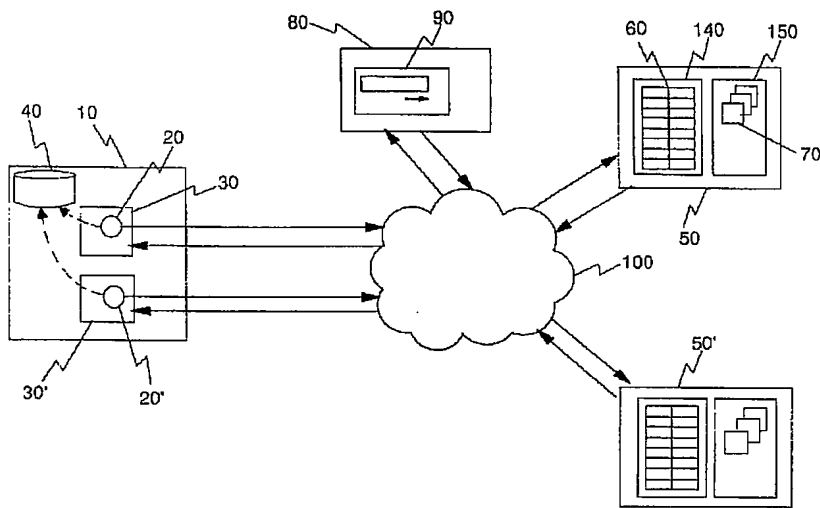
【図２】ソフトウェア・バージョン、及び或るバージョンから別のバージョンを形成するためのリソース及び前提のソフトウェア・ベンダのリストの一例である。

【図３】本発明の実施例に従ってアップデータ・コンポーネントの動作シーケンスを表す。

【図４】アップデータ・コンポーネントのオペレーションのシーケンスの一部分を更に示す。

【図５】アップデータ・コンポーネントのオペレーションのシーケンスの他の部分を更に示す。

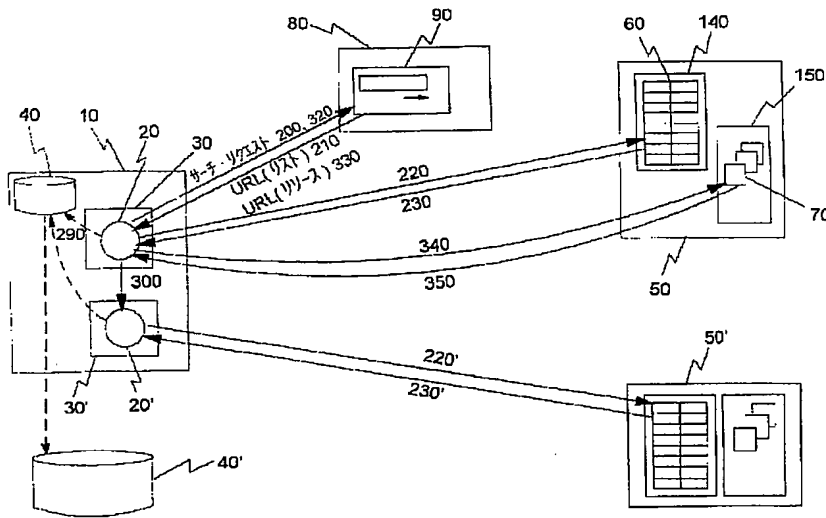
【図 1】



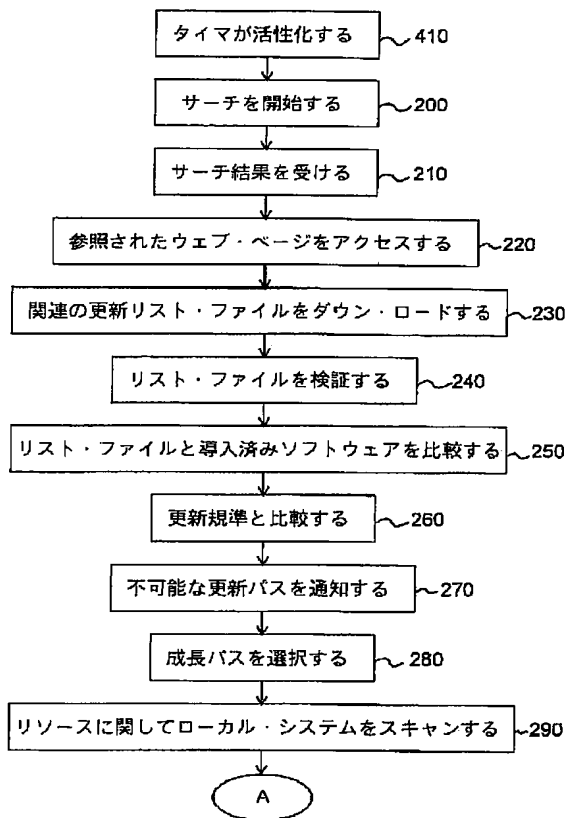
【図 2】

製品セット	更新リソース	前 提
ソフト製品 1 v1.0.0	—	オペレーティング・システム T3 v2.0
ソフト製品 1 v1.0.1	ソフト製品 1 に対するパッチ 1	オペレーティング・システム T3 v2.0
ソフト製品 1 v2.0.0	ソフト製品 1 に対するパッチ 2	オペレーティング・システム T3 v2.0
ソフト製品 1 v3.0.0	ソフト製品 1 v3.0.0 (置 換)	オペレーティング・システム T3 v2.0
ソフトゲーム 1 v1.0	—	オペレーティング・システム T3 v2.0
ソフトゲーム 2 v2.0	ソフトゲーム 2 に対するパッチ 1	オペレーティング・システム T3 v3.0
ソフトゲーム 3 v3.0	ソフトゲーム 2 に対するパッチ 2	オペレーティング・システム T3 v3.0

【図 3】



【図 4】



【図 5】

